The *impresso* system architecture in a nutshell[1]

Matteo Romanello[*1], Maud Ehrmann[*1], Simon Clematide[2], Daniele Guido[3]
[1] *École polytechnique fédérale de Lausanne (EPFL)*
[2] *Institut für Computerlinguistik der Universität Zürich*
[3] *Luxembourg Centre for Contemporary and Digital History (C²DH)*

1. Introduction

The decades-long efforts of libraries and transnational bodies to digitize historical newspapers holdings has yielded large-scale, machine readable collections of digitized newspapers at regional, national and international levels [1,2]. If the value of historical newspapers as sources for research in both academic and non-academic contexts was recognized long before, this "digital turn" has contributed to a new momentum on several fronts, from automatic content processing to exploration interfaces and critical framework for digital newspaper scholarship [3]. Beside the multiplication of individual works, hackathons and evaluation campaigns, several large consortia projects proposing to apply computational methods to digitized newspapers at scale have recently emerged (e.g. *Oceanic Exchanges*, *NewsEye*, *Living with Machines*).

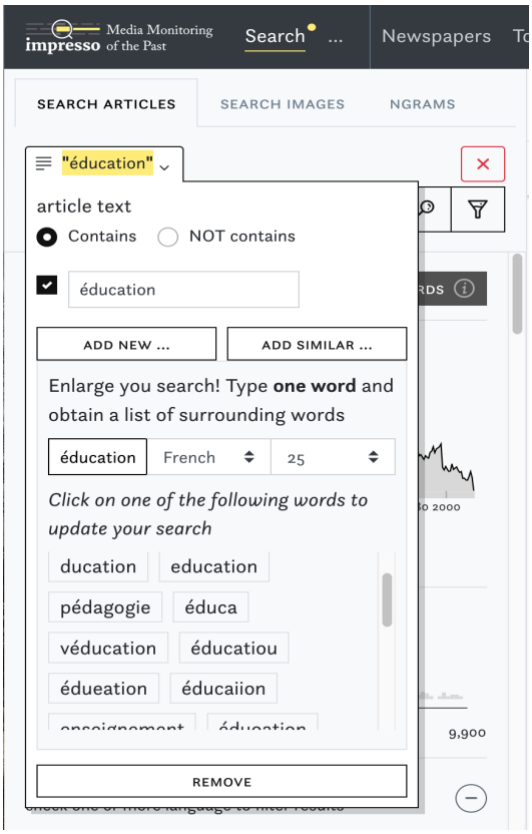
Among these initiatives, the project *impresso - Media Monitoring of the Past* tackles the challenge of enabling critical text mining of large-scale newspaper archives and has notably developed a novel newspaper exploration user interface. More specifically, *impresso* is an interdisciplinary research project in which a team of computational linguists, designers and historians collaborate on the semantic indexing of a multilingual corpus of Swiss and Luxembourgish digitized newspapers. The primary goals of the project are to improve text mining tools for historical text, to enrich historical newspapers with automatically generated data, and to integrate such data into historical research workflows by means of a newly-developed user interface. The *impresso app* is a full-fledged, in production newspaper interface with powerful search, filter and discovery functionalities based on semantic enrichments together with experimental contrastive views.

[1] This text was published in October 2020 in issue number 16 of the EuropeanaTech Insights dedicated to digitized newspapers and edited by Gregory Markus and Clemens Neudecker: https://pro.europeana.eu/page/issue-16-newspapers#the-impresso-system-architecture-in-a-nutshell

[*] Both authors contributed equally to this post.

Screenshot 1: Home page of the impresso newspaper application.

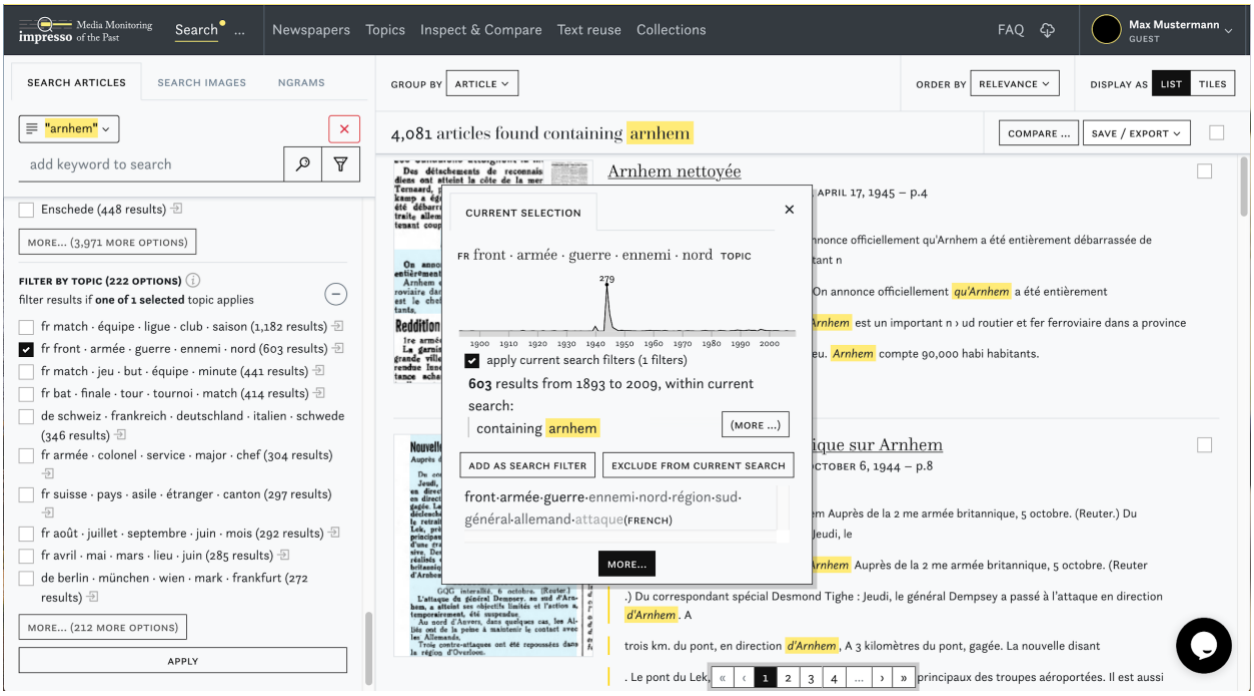
In order to situate the *impresso* app within the landscape of newspaper interfaces, we can recapitulate different newspaper interface generations identified by [4], based on the survey of 24 interfaces and the analysis of ca. 140 features: the first generation focuses primarily on making content available online, the second on advanced user interaction with the content, the third on automated enrichment, and the fourth on personalization and increased transparency. Fourth-generation interfaces are under development in research projects and the *impresso* one belongs to this group. Taking a closer look now, the impresso interface leverages a broad range of automatically generated semantic enrichments and allow manifold combinations of the following features: creation and comparison of user-generated collections (cf. screenshot 4 below); keyword suggestions (cf. screenshot 2 below); content filters based on topic models, named entities and text reuse (cf. screenshot 3); article recommendations; exploratory interfaces for text reuse clusters, n-grams and topics; image similarity search (cf. screenshot 5); visualisation of gaps, biases in the corpus and quality scores for OCR and entities. With co-design and experimentation among our core working principles, the interface was inspired by user feedback collected during multiple workshops, and motivated by the overarching goal to seamlessly shift between close and distant reading perspectives during the exploration of a large newspaper corpus. For further information on the project and the interface functionalities we refer the reader to the *impresso* clip and application guided tour. A more complete overview will be published in [5].



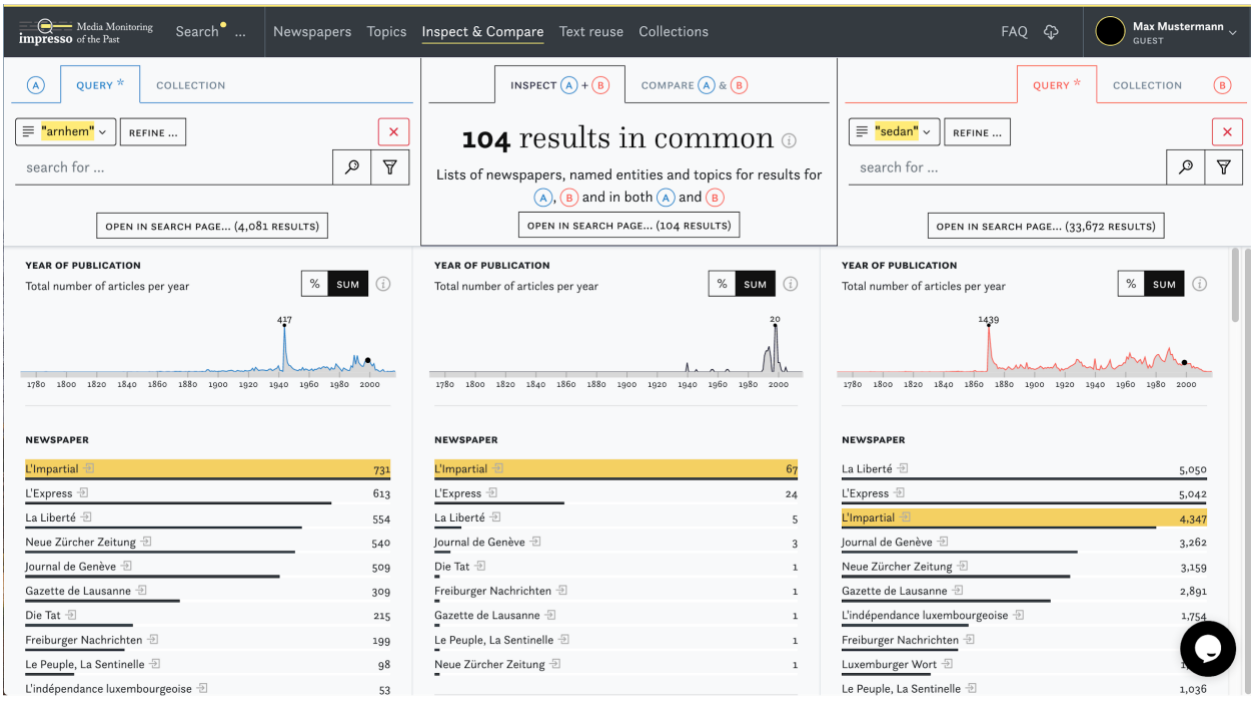
Screenshot 2 (Oct. 2020): keyword suggestion based on word embeddings, here for the word 'éducation'. Semantically-related words are suggested ('pédagogie') but also OCR mistakes ('éducatiou', 'véducation', 'éducaiion').

Our objective with this interface is, beyond enhanced access to sources which other portals also provide, to find how to best combine machine and human work. The *impresso* app targets first and foremost digital scholars and provides tools for the mediation with computationally enriched historical sources. In this sense, the overall purpose is less to enable the discovery of statistically relevant patterns, but rather to facilitate iterative processes characterised by searching, collecting, comparing and discovering which yield insights into historical sources and thereby inform further exploration. The *impresso* app should be seen as a research prototype, built in the context of a research endeavour and not as an infrastructure project.

That being said, building such an application requires the design and implementation of a solid system architecture. With designers, developers and computer scientists, the *impresso* team was well prepared for this, but had to face several challenges. Although technical considerations and operations compose the foundations of many cultural heritage-related projects and greatly contribute to shaping their outcomes, they are rarely under the spotlight. With this blog post we wish to do justice to these undertakings, and propose to briefly present the challenges we faced, the solutions we adopted, and the lessons we have learned while designing and implementing the *impresso* application.



Screenshot 3 (Oct. 2020): search filtering based on topics. Here the search "arnhem" can be filtered further via topics relating to either war or sport.



Screenshot 4 (Oct. 2020): Inspect and Compare view, here for the search "arnhem" and "sedan".

2. A complex construction site

The development of the *impresso app* was informed by an array of needs, constraints and activities stemming from different groups of actors, and often looked like a lively but also complex construction site. We consider the entirety of the *impresso* application, including backend (data storage, pre-processing and processing), middle-layer (API) and frontend. In the following we outline the main centers of influence which shaped this endeavor, whose elements might sometimes overlap.

1. Data *(or what are we working with)* corresponds to original data or derivative data. Original data is provided by libraries and archives and consists in our case of three types of objects: image scans, optical character recognition (OCR) and optical layout recognition (OLR) outputs, and metadata. Derivative data is the output of various processes applied to original data and corresponds to normalized original data and semantic enrichments of various kinds.

These data feature **characteristics** which often translate into needs:

- Original data is **dispersed** on various institutions' premises, which entails the need to physically acquire and store it (in order to process it), or to have a way to query it, typically via an API. In terms of system architecture and software design, this impacts the initial setup, with the need for storage facilities and/or integration of decentralized data access points, as well as the maintenance, especially with data updates in a distributed context (e.g. change of the IIIF URL scheme by a library).
- Original data have **different legal statements**, which impacts which and how different parts of the corpus can be used and shared. Beyond the administrative work of copyright clearance, this implies the definition of a data access policy as well as the implementation and management of user login reflecting different data access levels.
- Original data comes in **a variety of legacy formats,** which entails the need to normalize items (images, OCR outputs, metadata) towards a common format efficient for storage but mostly easy to manipulate and 'compute on' in distributed processing environments.
- Original data is often **noisy**, with respect to both its contents (imperfect OCR) and shape (missing, corrupted or inconsistent collection parts). This requires the development of robust text processing tools, as well as thoughtful and numerous data sanity checks.
- Original and derivative data correspond to **huge volumes**, with e.g. 70TB for the whole *impresso* original data, and more than 3TB of compressed textual data (stripped from unneeded information). Beyond storage, such volumes require distributed computing capacities (for processing) as well as hardware and software settings ensuring a good responsiveness.
- Original data can **grow**, when e.g. a new collection is planned for ingestion, which entails the need for scaling up capacities.

2. Actors/Stakeholders *(or who interact with the interface and/or data)* are diverse and this entails the consideration of various types of needs or interests. Actors include:

- Scholars, and particularly historians, who compose *impresso app*'s primary user group. Among many needs related to historical research (which were addressed throughout the project), one point which impacted most infrastructure and processes is the need for

transparency, especially concerning the gaps in the newspaper corpus. Besides paying special attention to the treatment of corpus 'holes' and inconsistencies, transparency also demands the ability to use versioning to indicate which version of a) the data and b) the platform was used at a given point in time.

- Libraries and data providers in general compose another group, interested in probing new ways to enhance their holdings, in testing their data access points (mainly IIIF et metadata APIs), and in benefiting from – and eventually recover – tools and semantic enrichments. In concrete terms, these interests translate into integration of external services, code and architecture documentation as comprehensive as possible, and derivative data serialization and packaging.
- Data scientists and NLP researchers, who are mainly after programmatic access to data, which requires a secured and documented API.

Besides continuous dialogue, the presence of various actors often requires different recipes to answer the same need, e.g. access to enrichments via, schematically, a user interface for scholars, an API for data miners, and dumps for libraries.

3. Activities *(or what do we do with the interface and/or data).* Abstracting away from objects at hand and actors, another perspective which helped formulating requirements corresponds to activities. Without going in too much details, we identified:

Data-related activities:

- search, access and navigate;
- research and study;
- transform, enrich, curate;
- cite.

System-related activities:

- store;
- compute;
- deliver;
- visualize.

Overall, data, actors and activities contribute a diverse set of requirements and compose the variegated landscape we evolved in while building the *impresso* app. If none of the questions, needs or requirements, taken in isolation, correspond to an insurmountable challenge, we believe that their combination introduces a substantial complexity. Challenges include conflicting requirements, with e.g. the need for both transparency and robustness, and conflicting "timing", with the need to develop while having the interface already used in production. Beyond this brief overview, the definition of these "centers of influence", the categorization of needs and their mapping to concrete requirements deserve further work which is beyond the scope of this post.

It is now time to dig into more concrete aspects.

3. The *impresso* system architecture

In this section we "dissect" the *impresso app* along three axes: data, system architecture, and processes.

Data

- As per storage, working copies of original data (images and OCR outputs) are stored on a centralized file storage service (NAS), with redundancy and regular snapshots and secured access reserved to project collaborators.

- Copyright and reuse status of data is managed at two levels. First, via a set of data sharing agreements between content providers and impresso partners (EPFL, C2DH, UZH). Second, users have to sign a non-disclosure agreement (NDA) to gain full access to the *impresso* collection. *Impresso* newspapers are either in the public domain and can therefore be used without restrictions (and accessed without login), or are still under copyright and can only be used for personal and academic use (login required). Rights are specified at a year level by data providers, and encoded at the article level in the *impresso* app. User NDA is accessible on the home page, and the terms of use documented in the app's FAQ.

- Original newspapers data come in a wide variety of OCR formats: 1) the Olive XML format (proprietary); 2) three different flavors of METS/ALTO; 3) an ALTO-only format; 4) the XML-based TETML format, output by the PDFlib TET (Text and Image Extraction Toolkit) which is used to extract contents from materials delivered in PDF format. These heterogeneous legacy formats are reduced to a "common denominator", namely a JSON-based schema, developed by the *impresso* project and openly released (see *Impresso* JSON schemas). Such a schema was designed to respond to the need for a simple, uniform, storage-efficient and processing-friendly format for the further manipulation, processing and enrichment of newspaper data.

System Architecture

- **Text data** are stored on a cloud-based object storage run by an academic network, accessible via the Simple Storage Service (S3) protocol, which is particularly suitable when distributed processes need read/write access to data. This S3-based storage is used both for the canonical data and for intermediate data that are the result of automatic processing.

- **Image data** are served by image servers which implements the International Image Interoperability Framework (IIIF) protocol. They are hosted either at the libraries' premises or on a project's image server. At the time of writing, the over 54.3 million-page images that are searchable via the *impresso* app are spread over four different IIIF-compliant image servers.

- **Indexing** of newspaper data (text, images and metadata) is powered by Solr, an open source indexing and search platform. We ingest into Solr both the canonical (textual) data as well as the output of enrichments (topic modelling, named entity recognition and linking, text reuse detection, etc.). The *Impresso* Solr instance contains several indexes (Solr collections) whose elements relate to each other, mainly on the basis of content item IDs (the basic unit of work, either article if OLR was performed, or page otherwise). A Solr plugin was developed to enable efficient numerical vector comparison.

- Finally, a **MySQL database** is used to store:
    a. metadata that are not indexed for faceted/full-text search (e.g. descriptive metadata about newspapers, issues and pages);
    b. user-related data such as login credentials, user-defined collections, etc.

Processes, or Data transformations. What lies *behind* the *impresso app* interface is a complex flow of processes that manipulate, transform, enrich and finally deliver *impresso*'s newspaper data to the frontend. These processes—most of the times transparent to end users—profoundly shape the data that can be searched and explored via the application:

- **Ingestion** is the process that consists in reading original data from the NAS storage and ingesting them into different parts of the backend, depending on their type. Text (OCR) is converted into *impresso*'s JSON canonical format defined in the JSON schemas, and subsequently stored on S3. At this step each content item receives a unique identifier. Images are either accessed directly via content providers' IIIF endpoints, or converted to JPEG2000 and ingested into the project's image server.

- **Rebuild** corresponds to the process of converting text-based data into different shapes, suitable for specific processes. These rebuilt data are transient as they can be regenerated programmatically at any time, and should be thought of as intermediate data that fulfill a specific purpose and whose shape is dependent on the process or tool that is supposed to act on them. For instance, as part of the ingestion process, canonical data are rebuilt into a JSON format that is optimized for ingestion into Solr.

- **Data sanity check** is meant to verify and guarantee the integrity of ingested data by checking e.g. the uniqueness of canonical identifiers used to identify and refer to newspapers data at different levels of granularity (issue, page, content item), or the fact that a page belongs to an issue (no orphaned items).

- **NLP enrichments**: original and unstructured newspapers data are enriched through the application of a series of NLP and computer vision techniques which add various semantic annotation layers. These enrichments are: language identification, OCR quality assessment, word embeddings, part-of-speech tagging and lemmatisation for topic modelling, extraction and linking of named entities, text reuse detection, image visual signatures. Since understanding the basics of each enrichment is essential for end users to be able to fully understand how to use the *impresso* app for their research, a wide array of pedagogic materials was created to this end [6,7], as well as extensive documentation in the application itself (i-buttons and FAQs).

- **Middle layer API**: It is a software component sitting between the frontend (user interface) and backend API that delivers JSON data to the front-end. It fetches data from different backend components (Solr, MySQL, image servers) and combines them. It also does some intermediate caching in order to speed up performances and enhance the application's responsiveness. It manages (via processing queues) asynchronous operations that are triggered by user actions (e.g. the creation of user collections) and stores the results of these operations to the backend.

All code is (being) released under the AGPL 3.0 license and can be found on impresso's GitHub organization page. Code libraries, processes and architecture will be documented in the "impresso cookbook" (in preparation). Derivative data are (being) released as described in [8].

4. Open challenges

As far as the system architecture is concerned, there remain two main open challenges that will need to be addressed by future research. The first one has to do with **dynamic data**, i.e. those enrichments such as topic modelling or text reuse that create "entities" that can be referenced via the *impresso* app. Topic modelling produces topics, each provided with a URI (e.g. https://impresso-project.ch/app/topics/tm-fr-all-v2.0_tp22_fr) and a dedicated topic page; similarly, text reuse yields clusters such as https://impresso-project.ch/app/text-reuse-clusters/card?sq=&clusterId=tr-nobp-all-v01-c111669150764. The problem is that these entities and their identifiers will disappear the next time the NLP processing of the corpus is executed (which happens every time new data are added to the corpus). Currently, the application does not support the co-existence of multiple versions of topic modelling or text reuse outputs, as this would have a substantial impact on the backend storage. Another possibility, which to date remains unexplored, is trying to align with one another successive versions of the same enrichment in order to support some kind of redirection mechanism.

The second open challenge lies in supporting **incremental updates** to the corpus. Currently, adding new or updated data triggers a complete reingestion and reprocessing of the entire corpus. This situation is relatively frequent as new material is acquired, and content providers may provide updated versions of already delivered content. However, given the multiple backends where data are stored and indexed, as well as the chain of NLP processing that is performed on top, supporting such incremental updates is far from being a trivial task from a system architecture point of view.

Notes & Acknowledgements

Authors warmly thank Lars Wieneke for his feedback, as well as the Swiss National Science Foundation (SNSF) for his support (Sinergia program, grant number CR-SII5_173719).

References

[1] Hildelies Balk and Conteh Aly. 2011. “IMPACT: Centre of Competence in Text Digitisation.” In Proceedings of the 2011 Workshop on Historical Document Imaging and Processing, 55–160. HIP ’11. New York, NY, USA: ACM. https://doi.org/10.1145/2037342.2037369

[2] Clemens Neudecker and Apostolos Antonacopoulos. 2016. “Making Europe’s Historical Newspapers Searchable.” In 2016 12th IAPR Workshop on Document Analysis Systems (DAS), 405–10. Santorini, Greece: IEEE. https://doi.org/10.1109/DAS.2016.83

[3] Mia Ridge, Giovanni Colavizza, Lauren Brake, Maud Ehrmann, Jean-Philippe Moreux and Andrew Prescott. 2019. “The Past, Present And Future Of Digital Scholarship With Newspaper Collections”. Multi-paper panel presented at the 2019 Digital Humanities Conference, Utrecht, July 2019. https://infoscience.epfl.ch/record/271329?ln=en

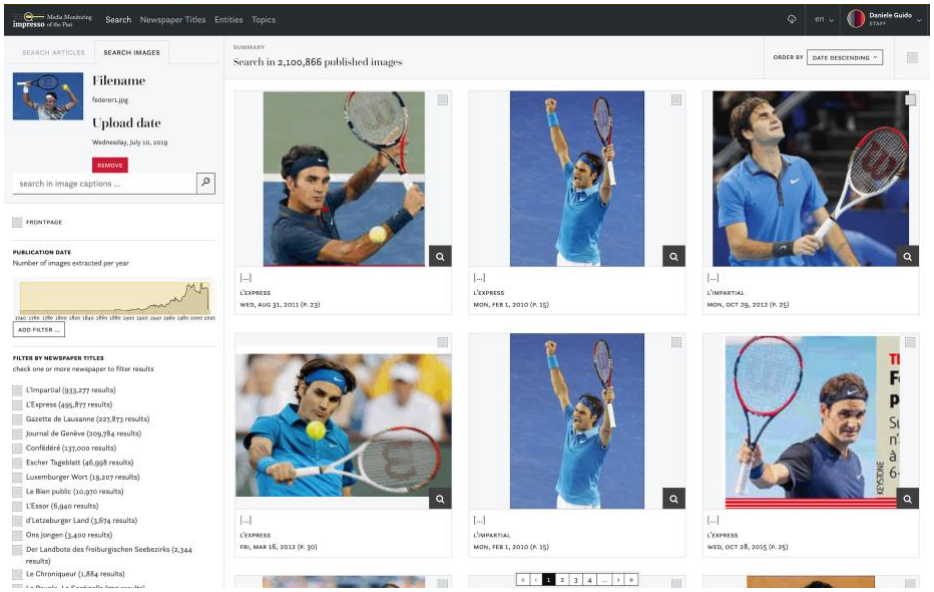
[4] Maud Ehrmann, Estelle Bunout, and Marten Düring. 2019. “Historical Newspaper User Interfaces: A Review”. In IFLA WLIC 2019 - Athens, Greece - Libraries: dialogue for change. https://doi.org/10.5281/zenodo.3404155

[5] Impresso team (2021, article in preparation). “Impresso: Historical Newspapers Beyond Keyword Search”.

[6] Estelle Bunout. 2019. “A guide to using collections of digitised newspapers as historical sources”, Parthenos Platform [link].

[7] Estelle Bunout, Marten Düring and C2DH. 2019. “From the shelf to the web, exploring historical newspapers in the digital age”. https://ranke2.uni.lu/u/exploring-historical-newspapers/

[8] Maud Ehrmann, Matteo Romanello, Simon Clematide, Phillip Benjamin Ströbel, and Raphaël Barman. 2020. “Language Resources for Historical Newspapers: the *Impresso* Collection”. In Proceedings of The 12th Language Resources and Evaluation Conference. https://www.aclweb.org/anthology/2020.lrec-1.121



Screenshot 5 (Oct 2020) Example of visual search results with a user-uploaded image.